基于适应性的网格数据库中子查询的节点优化调度*)

胡乃静[1,2] 赵 亮[1] 胡金华[3]
(复旦大学计算机与信息技术系 上海 200433)[1] (上海金融学院信息管理系 上海 201209)[2]
(东华大学计算机学院 上海 200051)[3]

摘 要 网格数据库是数据库技术和网格技术相结合后新的研究领域,网格的动态变化特性对数据库查询优化技术提出了适应性的要求。本文提出了基于Petri网描述的子查询计划模型TNSN,通过扩展子查询及其节点的数据关联关系的描述,建立了子查询进行适应性优化调度的查询计划模型;进一步提出了考虑变化的参数在内的耗费估算模型,并在TNSN和耗费模型的基础上提出了适应性优化算法,保证了查询处理过程中可以根据网格参数的变化情况对查询进行适应性调整,最后给出了实验验证。

关键词 网格,数据库,查询,适应性

Adaptive Node Schedule of Query Optimization in Grid Database

HU Nai-Jing[1,2] ZHAO Liang[1] HU Jin-Hua[3]
(Department of Computer and Information Technology, Fudan University, Shanghai 200433)[1]
(Department of Information Management, Shanghai Finance University, Shanghai 201209)[2]
(College of Coputer, Donghua University, Shanghai 200051)[3]

Abstract Grid database is a new research area combined the database techniques and Grid techniques, traditional database query and optimization methods was asked to be adaptive because of the dynamic properties of Grid environment. The paper brings forward a query plan execution model TNSN based on Petri net to describe the node scheduling and execution efficiency of sub-queries, to analysis the interrelation and query consistency of TNSN nodes, and guarantee the validity of reschedule policy of TNSN. The paper further puts forward a query-expend model of TNSN, and also gives the dynamic and optimization tuning algorithm used by the query execution plan. Finally the paper tests the models and algorithms.

Keywors Grid, Database, Query, Adaptive

1 前言

1.1 介绍

网格数据库[1]是数据库技术和网格技术相结合后新的研究领域。随着符合GGF标准的Globus工具包[4]以及网格数据库接口标准OGSA-DAI[5]中间件的发展,越来越多的研究关注网格数据库的查询处理,如PolaR*[6]和OGSA-DQP[7]是采用网格技术提供数据库查询的两个研究项目。

与传统分布式数据库不同,网格数据库中由于网格具有动态的特性,查询处理中存在局部代价参数不可得、不精确、不完全或变化的情况,使得传统的查询优化技术不能满足网格数据库中的查询优化要求[2],需要采取适应性的查询处理(adaptive query processing, AQP)[3],即在查询计划执行期间,在网格环境参数发生变化的情况下,对查询计划中尚未进行的字查询进行适应性的动态优化调整,以保证最好的效率。

1.2 相关研究

相关的网格数据库查询处理的研究中,PolaR*[6]和OGSA-DQP[7]采用的是静态查询处理方式,按照每一个到达的查询依顺序进行查询的翻译、优化和执行;文[8]中的SwAP研究了网格环境参数的动态获取,通过设置Eddy进行监控,取得相关的工作负载、传输速率等信息。GrADS[10]以及Condor[11]采用基于DAG的模型描述子查询计划,支持子查询之间简单的前继、后继的依赖关系及其子查询的执行的耗费描述;文[9]对子查询在网格中的节点分配优化进行了研究,采用min-min的调度算法,考虑局部分派和全局执行耗费的影响,将子查询按照计算任务的处理模式进行启发式优化调度。

1.3 相关分析

上述的有关研究中,文[6,7]作为早期项目采用静态的查询处理方式,无法满足动态网格环境的需要。文[9~11]中将子查询按照网格中计算任务的情况进行建模和适应性的优化调度,具有以下不足:

·子查询与单纯的计算任务有不同,计算任务主要使用节点的内存、CPU等资源以完成计算,只要能够满足计算需求,可以调度在任何一个节点执行;而子查询不同,主要使用节点数据库中的数据资源,子查询与节点之间存在着数据关联关系,因此子查询不能像计算任务一样进行任意调度,否则只会降低效率。文[9]中的调度算法则忽略了子查询与执行节点之间的数据关联关系。

·计算任务通常功能性较强,彼此之间相对独立,数据依赖关系简单;而子查询本身由全局查询分解而成,彼此之间的数据依赖关系紧密,存在着丰富的关联关系。文[10,11]中的

*)上海市青年科技启明星计划资助(05QMB1430)。**胡乃静** 博士,副教授,研究方向:数据库。

DAG模型式，对子查询之间的相互关系描述能力不强，不支持子查询之间的顺序、并发以及其它的关联关系描述，因而其模型对子查询的适应性进化支持力度不足。

1.4 本文研究内容

本文研究网格数据库中子查询在节点的适应性优化调度，分为两部分内容：

· 基于Petri网描述的子查询模型TNSN。基于DAG子查询计划描述的依赖关系，引入了子查询的执行节点以及子查询的执行时间，采用Petri网的描述方式，建立了能够描述子查询及其执行节点调度的模型TNSN，作为适应性优化调度的基础；

· 查询耗费模型及其适应性优化算法。在TNSN的基础上，建立了子查询的数据通讯耗费和子查询的数据处理耗费为主的耗费表达模型，进而确立了不同网格环境参数下子查询的最优耗费计算模型，并根据耗费计算模型设计适应性的调度算法，以进行适应性调整。

2 全局查询及其有关的基本定义

网格数据库中查询处理的模式主要有基于全局虚拟数据视图VO(Virtual Organizations)和基于P2P(Peer-to-Peer)两种。基于VO的查询处理主要将针对网格数据库的查询(称之为全局查询)分解为子查询，并将子查询分派到相应的节点上进行查询处理。因此，基于VO的模式其查询建立在VO的基础上，VO则是通过抽取各节点数据库中的模式并集成所形成的一个全局统一的数据视图，其中存储的是各节点的模式元数据，而实际的物理数据分在各节点数据库中。下面给出基于VO查询的有关基本定义。

定义1 全局虚拟数据视图VO 考虑n个节点组成的网格数据库，节点集合$P=\{P_1,P_2,\cdots,P_n\}$，假定节点$p_i(1\leqslant i\leqslant n)$的属性集为$v_i(1\leqslant i\leqslant n)$，则VO定义为全体属性集合的并集$U=\sum_{i=1}^{n}v_i=v_1\cup v_2\cup\cdots\cup v_n$。

定义2 全局查询 根据传统的数据库查询，一个网格数据库查询需要根据VO分解为针对各节点数据库的子查询操作的查询树，令属性集$u_i\subset U$，定义$q_1=\theta(u_1),\cdots,q_i=\theta(u_i)$。其中$q_1,\cdots,q_i$为查询分解树的叶子节点，则定义全局查询为$Q(U)=\{q_1,q_2,\cdots,q_i,\cdots,q_k\}$。

定义3 子查询 定义全局查询$Q(U)=\{q_1,q_2,\cdots,q_i,\cdots,q_k\}$中的叶子节点查询$q_i=\theta(u_i)(1\leqslant i\leqslant k)$为全局查询中的子查询。

定义4 节点子查询 子查询q_i最终被调度在某个节点数据库p_j上的执行查询操作，称之为节点子查询(q_i,p_j)。明显地，节点子查询(q_i,p_j)分为集中式与分布式两类：

· 集中式。节点数据库p_j的数据集合能够满足集合子查询q_i，不需要访问其它节点；

· 分布式。类似于分布式查询，q_i除执行节点p_j外，需要访问其它的节点$p_1,\cdots,p_j,\cdots,p_k$。

3 基于Petri网的子查询模型

3.1 子查询模型TNSN的定义

传统查询计划的描述为DAG图，但形式化描述能力不足。Petri网具有很好的形式化基础，采用Petri网进行描述，在查询计划中引入节点子查询，同时引入执行时间的概念，描述各个子查询在其调度节点的执行时间，从而形成了网格数据库中描述查询计划的时序节点子查询网TNSN(Time Node Subquery Net)。

定义5 TNSN TNSN=(P,Q,F,T)

(1)Q为子查询，分为FQ,RQ两类，代表子查询执行的两个阶段，即将一个子查询q_i描述为前端和后端两个阶段：FQ定义查询q_i分为前端部分，表述为$f(q_i)$，代表查询开始，图中用$\boxed{q_i}$表示；RQ为尾端部分，代表查询结束，表述为$r(q_i)$，图中用$\boxed{q_i}$表示。

(2)P代表节点数据库，用○表示，○标志节点数据库的编号或?，置于FQ的前方，代表子查询分派在该节点数据库上。

· 若○内标志节点数据库的编号，则表示子查询Q执行的节点数据库位置。

· 若○内标志?，则表示其后的子查询需要根据条件进行选择，执行节点在执行的时候才能确定。

(3)F为流关系，分为FT,FF两种。FT用实线表示，流向为子查询到另一个子查询的执行节点，代表一个子查询向另一个子查询执行节点的数据输出；FF用虚线表示，一类由节点流向节点，表示该节点向另一个节点存在数据传输的关系。

(4)T为时序关系，表明节点子查询完成所需要的时间耗费，分为两类：一类为传输时间，在网中由P指向Q，表明分布式子查询在节点P完成查询所需要的数据传输时间；一类置于$r(q_i)$之前，表示在该节点完成子查询所需的时间损耗。

例1 图1所示为TNSN网，其中子查询q_3和q_4是选择关系，即根据子查询q_1的执行结果进行选择；子查询(q_2,q_5)与(q_1,q_3,q_4,q_6)之间是并发的执行关系，依据文[9]中的CPR调度算法，全局查询q所需要的时间为：$q=0+13+3+1+1=18$。

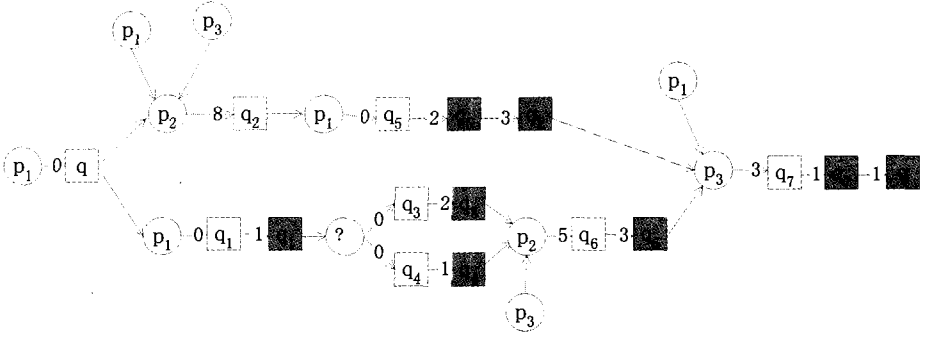

图1 TNSN例图

3.2 TNSN的子查询相互关系

在TNSN子查询的相互关系中，除了传统的前继、后继的依赖关系外，还存在着子查询的激发关系，即一个前继子查询在执行中，需要启动其它节点后继子查询，后继的子查询完

成并返回结果后，前继子查询继续执行。下面给出相互关系定义。

定义 6 激发关系 两个子查询(q_i,q_j)是激发关系，当 q_i 调用 q_j 的执行，q_j 完成后返回数据至 q_i，q_i 继续执行，完成查询工作。定义 q_i 为 q_j 的父查询，q_j 为 q_i 的子查询，表示为 →，即

$$q_i \rightarrow q_j = \{(q_i, q_j) \mid q_i \in Q(U) \wedge q_j \in Q(U) \wedge q_i \neq q_j, 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant n\}$$

定义 7 依赖关系 两个子查询[q_i,q_j]是依赖关系，q_j 在 q_i 结束后才执行，即 q_j 依赖于 q_i 执行结果，如 q_j 的输入为 q_i 的输出；定义 q_i 为 q_j 的前继查询，q_j 为 q_i 的后继查询。表示为⇒，即

$$q_i \Rightarrow q_j = \{[q_i, q_j] \mid q_i \in Q(U) \wedge q_j \in Q(U) \wedge q_i \neq q_j, 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant n\}$$

例 2 图 1 中，存在如下相互关系：

- $q_2 \rightarrow q_5$，即 q_5 由 q_2 激发；
- $(q_2, q_6) \Rightarrow q_7$，即 q_7 依赖于 q_2、q_6。

3.3 TNSN 的适应性优化调度分析

由于各节点的数据资源和处理能力不同，子查询不同的执行节点调度会产生不同的查询效率。在网格数据库中参数动态变化的情况下，如网络传输效率变化、节点处理速度变化等情况时，子查询原来选择的最优节点可能不再是最优的。因此，需要在网格环境的参数变化时，对子查询模型中尚未处理的子查询进行适应性的优化调度，重新选择最优的执行节点，以提高整个查询效率。

TNSN 中，节点子查询分为集中式和分布式两类。集中式由于在本地节点完成查询，类似于传统的集中式数据库查询，不存在数据传输以及其它节点的数据对其的影响，不需要考虑节点的适应性调整。而分布式节点子查询在不同的节点中执行的效率不同，在参数发生变化时，如网络传输效率下降、最优节点的处理能力下降等都可能会导致分布式子查询的最优节点发生变化，重新核定最优的执行节点会提高查询的效率，因此，本文中的适应性的优化调度将围绕着 TNSN 中的分布式子查询的执行节点的动态调度进行。

4 适应性的耗费估算及其优化调度算法

4.1 耗费估算模型

4.1.1 耗费表达模型

设分布式子查询 q_i 所需要的物理数据 u 分散在 $p_1, \cdots, p_k$ 等节点数据库中的数据集合 $\omega_1, \cdots, \omega_k$ 上，则发生在节点 p_j 上的时间耗费 $T(q_i, p_j)$ 为：其它节点向 p_j 传输所需数据的通讯耗费 T_{comm} 和在该节点完成查询处理 θ 操作（选择、投影以及连接）的时间耗费 T_{que}。表达式为

估算时间：$T(q_i, p_j) = T_{comm} + T_{que}$ (1)

- 通讯耗费 T_{comm}。T_{comm} 主要为 $p_1, \cdots, p_K$ 节点将数据集合 $\omega_1, \cdots, \omega_k$ 传输到 p_j 节点的时间损耗，与传输数据的数据量成正比，与网络的带宽成反比。设节点 p_m 与 p_j 之间的带宽为 $BW_{(m,j)}$，则 p_m 与 p_j 之间的传输时间耗费 $t_{(m,j)}$ 可描述为：$t_{(m,j)} = \frac{\omega_m}{BW_{(m,j)}}$，则

$$T_{comm} = \frac{\omega_1}{BW_{(1,j)}} + \cdots + \frac{\omega_m}{BW_{(m,j)}} + \cdots + \frac{\omega_k}{BW_{(k,j)}} = \sum_{i=1}^{k} \frac{\omega_i}{BW_{(i,j)}} \quad (2)$$

- T_{que} 的处理耗费。T_{que} 反映了该节点对于数据集合 u_i 执行有关 θ 操作的时间耗费，与节点的性能参数有关，如 CPU、磁盘 I/O 等机器性能。设节点 p_j 的处理能力定义综合参数为 pro_j，则

$$T_{que} = \frac{u}{pro_j} \quad (3)$$

- 总体时间耗费。由式(3)和式(4)可得，子查询 q_i 在节点 p_j 上的时间耗费 $T(q_i, p_j)$ 为

$$T(q_i, p_j) = T_{comm} + T_{que} = \sum_{i=1}^{k} \frac{\omega_i}{BW_{(i,j)}} + \frac{u}{pro_j} \quad (4)$$

4.1.2 模型的计算表达转换

为便于耗费模型的计算实现，对式(4)的耗费模型做进一步的转换，根据式(4)可求得子查询 q_i 在 $p_1, \cdots, p_k$ 等节点数据库中执行的各自的时间耗费。为描述方便，令

$$T(q_i, p_j) = A_j; \frac{1}{BW_{(k,j)}} = C_{jk}; \frac{1}{pro_j} = \psi_j$$

则式(4)可进一步表述为

$$A_j = \sum_{k=1}^{h} C_{jk}\omega_k + \psi_j u_i = (C_{j1}\omega_1 + C_{j2}\omega_2 + \cdots + C_{jj}\omega_j + \cdots + C_{jh}\omega_h + \psi_j u) \quad (5)$$

C_{jj} 表示节点向自己的数据传输带宽，值为 0。同理，构建 q_i 在 $p_m, \cdots, p_z$ 各个节点的耗费，为

$$\begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_k \end{pmatrix} = \begin{pmatrix} C_{11}\omega_1 + \cdots + C_{1j}\omega_j + \cdots + C_{1k}\omega_k + \psi_1 u \\ C_{21}\omega_1 + \cdots + C_{2j}\omega_j + \cdots + C_{2k}\omega_k + \psi_2 u \\ \cdots \\ C_{k1}\omega_1 + \cdots + C_{kj}\omega_j + \cdots + C_{kk}\omega_k + \psi_k u \end{pmatrix} = \begin{pmatrix} C_{11} & \cdots & C_{1k} & \psi_1 \\ C_{21} & \cdots & C_{2k} & \psi_2 \\ \cdots & \cdots & \cdots & \cdots \\ C_{k1} & \cdots & C_{kk} & \psi_k \end{pmatrix} \cdot \begin{pmatrix} \omega_1 \\ \cdots \\ \omega_k \\ u \end{pmatrix}$$

根据 C_{ij} 的应用意义可知，从节点 i 到节点 j 的网络带宽与从节点 j 到节点 i 的网络带宽是相同的，即 $BW_{(i,j)} = BW_{(j,i)}$，因此 $C_{ij} = C_{ji}$，同时 $C_{jj} = 0$，可进一步化为

$$\begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_k \end{pmatrix} = \begin{pmatrix} 0 & C_{12} & \cdots & C_{1k} \\ C_{12} & 0 & \cdots & C_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ C_{1k} & C_{2k} & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_k \end{pmatrix} + u \begin{pmatrix} \psi_1 \\ \psi_2 \\ \cdots \\ \psi_k \end{pmatrix} \quad (6)$$

4.2 适应性的优化调度算法

4.2.1 优化分析

式(6)中，C_{ij} 与网络带宽有关，在网格环境的动态变化中 C_{ij} 的值也会随着网络的情况发生变化；ψ_j 反映了节点的处理能力，随着节点的任务变化，ψ_j 也会发生调整；ω_i 与 u 为待处理的各项数据，不会随网格环境的变化而变化。因此，分析优化调度如下：

- 采用文[8]的 eddy 测试方法可测得当前网格环境下的 C_{ij} 与 ψ_j 的值，并代入式(6)进行计算，求得最低的 A_j，以判定当前值下的最低耗费节点 j，寻找到一个最优的节点调度。

- 如果该节点调度的耗费优于原有的节点调度耗费，则将新的该节点调度耗费带入由 CPR 确定的耗费模型中。若新确定的耗费模型 CPR 值小于原有的，则调整查询计划，将该子查询的查询节点修订为新确定的节点，进行适应性调整。否则，不对查询计划进行调整。

4.2.2 适应性优化算法描述

根据 4.2.1 的优化分析，确定适应性优化调整算法如下。

输入：查询计划 $TNSN_{old}$、监测的各节点的耗费参数 S[]、$TNSN_{old}$ 的 CPR 时序值 T_{old}

输出：调整的查询计划 $TNSN_{new}$

算法：

```
1. begin
2.    do while true
3.       next_conveger(q_i, P, P_N)
4.       if p=null goto 15
5.       select(P, P_N, P_[])
6.       if P_[]=null goto 3
7.       compute_min(P_[], S[], p)
8.       if p=P //节点不发生变化,then goto 3
9.       else
10.            compute_cpr(p, T_new)
11.            if Tnew<Told move(p) //判定新节点是否更
                 优,若是则更改
12.            goto 3
13.      end if
14.   end do
15. end
```

nextconveger(q_i)函数主要用于在 TNSN 中，确定子查询 q_i 的查询一致点 P 以及该处的查询子网 P_N；select(P, P_N, $P_{[]}$)从封闭子网 P_N 中查找出需要在多个节点之间传输数据的子查询及其节点，并将数据输入 $P_{[]}$ 中；compute_min($P_{[]}$, S[], p)采用式(6)计算在新的参数 S[]下新的耗费最低的节点 p；compute_cpr(p, T_{new})采用 CPR 算法计算在新的节点更改下新的 CPR 时序值 T_{new}。

5 实验

本文以具有适应性特征的算法与文[6]中的 Polar 静态查询处理算法以及仅考虑节点计算能力进行子查询调度的文[9]中的 mini-mini 算法进行分析比较。搭建测试环境如下。

5.1 试验环境

搭建局域网的网格数据库环境，采用 Linux 操作系统、Glbous 3.0 以及 OGSA-DAI3.0 互连数据库。设有三个节点数据库 P_1，P_2，P_3，节点之间通过 100M 的局域网互连。节点 P_1 的数据库模式为 R，节点 P_2 的模式为 S，节点 P_3 的模式为 T，各配置信息如表 1 所示。

表 1 环境信息表

节点	CPU	内存	数据库	数据模式	元组数
P_1	P4 2.6 G	1024M	Oracle	R	10000
P_2	P3 1.0 G	256M	access	S	1000
P_3	P4 2.4 G	512M	SQL Server	T	5000

构造子查询 $q_1(P_3)$、$q_2(P_3)$、$q_3(P_1)$、$q_4(P_2)$、$q_5(P_2)$。$q_1(P_2)$表示初始化的查询计划模型中子查询 q_1 的执行节点为 P_2；其中 q_2 是集中式子查询，q_1、q_3、q_4、q_5 是分布式子查询；假设子查询 q_1、q_2、q_3、q_4、q_5 按顺序执行。子查询执行期间，构造网格的变化环境如下：

- 子查询 q_1 执行期间网格环境保持不变；
- 子查询 q_2 执行期间，增大网络的阻塞，降低网络的通讯能力；
- 子查询 q_3 执行期间，增加节点 P_2 的任务，降低其处理能力；
- 子查询 q_4、q_5 执行期间，保持变化后的情况不变。

5.2 测试结果及其分析

测试数据如表 2 所示。

表 2 测试数据

序号	子查询	Polar 静态调度			Mini-mini 任务调度			本文适应性调度		
		执行节点	查询耗费	通信耗费	执行节点	查询耗费	通信耗费	执行节点	查询耗费	通信耗费
1	q_1	P_3	16	20	P_2	16	20	P_1	16	20
2	q_2	P_3	7	0	P_3	7	0	P_3	7	0
3	q_3	P_1	112	93	P_1	112	93	P_3	135	47
4	q_4	P_2	171	39	P_3	59	62	P_1	65	40
5	q_5	P_2	74	27	P_3	65	57	P_2	74	27
总计			380	179		259	232		297	134
			559			491			431	

从表 2 可以看出，polar 的静态调度算法在网格环境发生变化的时候，依旧保持不变，而 mini-mini 和随机调度则不同：

- 子查询 q_2 执行期间，网络的传输效率下降；在适应性调度算法中，随后执行的 q_3 在执行节点上发生了变化，调整为 P_3；而由于 P_1 节点的处理能力没有发生变化，因此 mini-mini 算法不做调整，因而在 q_3 子查询处适应性的效率较好。
- q_4 子查询由于 P_2 节点性能下降，mini-mini 重新调整为一个处理最快的节点 P_3，而适应性则选择一个综合性能最好的 P_1 节点。
- 子查询 q_5 由于网络传输和 P_2 节点处理能力下降，mini-mini 则选择处理能力最好的 P_3，而适应性则选择了综合效率最高的节点 P_2。

表 2 的结果显示，mini-mini 算法的查询耗费是最低的。但由于忽略了数据传输方面，因此综合效率最好的是适应性调度算法，而静态调度算法则在网格环境发生变化的情况下依旧保持不变，效率是最低的。

结束语及未来研究工作 网格数据库中网格环境的动态变化，存在局部代价参数不可得、不精确、不完全或变化的情况，使得传统的查询优化技术不能够完全满足网格数据库的优化要求。本文在 Petri 网形式化描述的基础上，提出了对查询计划的时序 Petri 的描述方法 TNSN，从而能够对查询计划中的子查询的节点调度及其效率进行描述；本文对 TNSN 从子查询之间的激发关系、依赖关系等进行分析，提出了封闭查询子网及其查询点的定义，从而保证了 TNSN 的动态优化的正确性，并在此基础上给出了查询耗费的模型及其动态优化调度的 CPR 算法，最后给出了实验验证及其分析，从而实现了网格数据库中查询的适应性优化。

本文虽然实现了查询计划执行过程中根据环境参数的变化进行优化，但仅仅实现了查询计划中尚未调度的部分子查询的调度优化，对于正在进行中的子查询的动态优化未予考虑。所以，根据环境参数的变化，对正在实施中的子查询的动态优化将是下一步的研究工作。

参 考 文 献

1 王珊，张坤龙. 网格环境下的数据库系统. www.chinagrid.net, 2005

2 Smith J, Gounaris A, Watson P, et al. Distributed Query Processing on the Grid. In: Proceedings of Grid Computing 2005, Springer, LNCS 2536, 2005. 279～290

（下转第 109 页）

程的文件映射对象建立视图，可以在它自己的虚拟地址空间建立完全一样的文件视图。这能使进程共享数据。任何有文件映射名字或句柄的进程都能建立文件视图[8]。

如果两个进程需要共享与文件无关的内存块，则一个进程必须使用 CreateFileMapping 函数指定（HANDLE）0xFFFFFFFF 而不是现有文件句柄。对应的文件映射对象从操作系统页面文件访问内存。共享操作系统页面文件的进程必须就在共享命名文件时通过使用 MapViewOfFile 或 MapViewOfFileEx 函数建立文件视图。

在该系统中运用共享内存关键是内存的格式化，即：格式化改内存为需要数据类型，并用一种合适数据结构管理数据。

3.5 共享内存的格式化

3.5.1 循环队列

通过共享内存能够成功实现进程间数据传递，特别适合大批量数据传递。但仅仅使用该共享内存来存取数据，不能满足系统实时数据传输要求，会出现数据覆盖或重复。该系统采用循环队列管理数据，以实现读写进程间的透明性，即：写进程不受读进程速度影响，可以透明地写内存。循环队列的首尾指针需要保存，并在读写程序中都能看到实时变化，故将首尾指针也保存在共享内存中。

3.5.2 多种数据类型

LabVIEW 采集程序的输出为 short int 二维数组、double 一维数组、char 二维数组和 int 一维数组。系统共申请了四块内存分别对应采集数据的四种数据类型，实现多种数据类型传输。

3.6 输入输出函数设计

DLL，设计了提供给 VC 信号处理程序和 LabVIEW 采集程序用于读写的函数。为了易于修改采集程序，对应每种输入数据分别设计输入输出函数。

采集程序有一个采样数组（Number of Samples）的概念。采样数组是采集程序的缓存，当采集数据将采样数组存满后再写入 DLL，即采集程序按数组形式写入。相应地，读取函数也按数组来读取，即读写数据都以数组为单位。

3.7 采集程序的修改

采集程序只能将采集数据存盘，故需要修改之后才能把数据传递到 DLL 中。修改 LabVIEW 采集程序的主要工作是：在存盘之前进行拦截，将数据传递到 DLL 中。使用 LabVIEW 提供的 CLF 节点调用 DLL。在采集程序中一共加入了五个 CLF 节点，分别对应需要传递的数据。

4 实验结果

将该实时数据传输技术用于脑机接口实验中。诱发电位信号放大和数据采集采用 BioSemi 公司生产的 Active One 生理信号采集系统，采集程序采用 LabVIEW 编写的生理信号采集程序，刺激器和信号处理程序都用 VC 开发。系统中，采集程序和信号处理程序调用 shared2DDll. dll 对应的输入输出函数。实验结果表明，运用文中的实时数据传输技术，能够很好地满足设计要求，实时地将采集数据传递到信号处理程序中，并将结果反馈给测试者。

结论 采用文件映射的共享内存方法可以很好地进行大批量数据传输，并且可以按需要格式化内存，能够很好地满足 BCI 系统实时要求。在以后的应用中，还需要更好地研究数据的读写同步机制，以避免在运行时间过长时出现数据等待和覆盖。

参 考 文 献

1 Wolpaw J R, Birbaumer N, McFarland D J. Brain2Computer interface for communication and control [J]. Clinical Neurophysiology, 2002, 113(6): 767～791

2 Wolpaw J R, Birbaumer N, Heetderks W J, et al. Brain- computer interface technology: A review of the first international meeting [J]. IEEE Transaction on Rehabilitation Engineering, 2000, 8 (2): 164～173

3 杨立才，李佰敏，李光林，贾磊. 脑机接口技术综述[J]. 电子学报，2005，33(7)：1234～1241

4 何庆华，彭承琳，吴宝明. 脑机接口技术研究方法[J]. 重庆大学学报，2002，25(12)：106～109

5 何庆华，彭承琳，等. 脑机接口视觉刺激器的研究[J]. 中国临床康复，2004，8(11)：2060～2061

6 欣力，等译. Microsoft Corporation. Microsoft Win32 程序员参考大全(一)[M]. 北京：清华大学出版社，1994

7 杨乐平，李海涛，等. LabVIEW 高级程序设计[M]. 北京：清华大学出版社，2003

8 欣力，等译. Microsoft Corporation. Microsoft Win32 程序员参考大全(二)[M]. 北京：清华大学出版社，1994

（上接第 98 页）

3 Babu S, Bizarro P. Adaptive query processing in the looking glass. In: CIDR, 2005. 238～249

4 Raman V, Narang I, Crone C, et al. Services for Data Access and Data Processing on Grids. In: GGF Document GFD. 14, Global Grid Forum, 2003

5 Anjomshoaa A, et al. The Design and Implementation of Grid Database Services in OGSA-DAI. In: Proceedings of UK e-Science All Hands Meeting, Nottingham, September 2004

6 Smith J, Sampaio S, Watson P. The design, implementation and evaluation of an odmg compliant, parallel object database server. Distributed and Parallel Databases, 2004, 16(3): 275～319

7 Alpdemir M N, Mukherjee A, Paton N W, et al. Servicebased distributed querying on the grid. In: Proceedings, ICSOC 2003, First International Conference, Trento, Italy, Springer, December 2003. 467～482

8 Zhou Yongluan, Ooi Beng Chin, Tan Kian-Lee, et al. An adaptable distributed query processing architecture. Data Knowl Eng, To appear, 2005

9 Blythe J J, Deelman S, Gil E, et al. Task scheduling strategies for workflow-based applications in grids. Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on May 2005, 2: 759～767

10 Dail H, Sievert O, Berman F, et al. Scheduling in the grid application development software project. In: Grid resource management: state of the art and future trends. Kluwer Academic Publishers Group, 2003

11 Thain D, Tannenbaum T, Livny M. Condor and the grid. In: Grid Computing: Making the Global Infrastructure a Reality, JohnWiley & Sons Inc, 2003